Excel for Cost Engineers

Abstract: Excel is a powerful tool with a plethora of largely unused capabilities that can make the life of an engineer cognizant of them a great deal easier. This paper offers tips, tricks and techniques for better worksheets. Including the use of data validation, conditional formatting, subtotals, text formulas, custom functions and much more. It is assumed that the reader will have a cursory understanding of Excel so the basics will not be covered, if you get hung up try Excel's built in help menus, or a good book.

Excel is more than the cornerstone that supports the cost engineer—more often than not, it is the entire toolbox. A powerful tool, Excel has largely unused capabilities that can make the life of a cost engineer much easier. This chapter covers some of these tools. This discussion assumes that the reader has a cursory understanding of Excel.

Data Validation

Figure 1 illustrates the Data Validation tool.

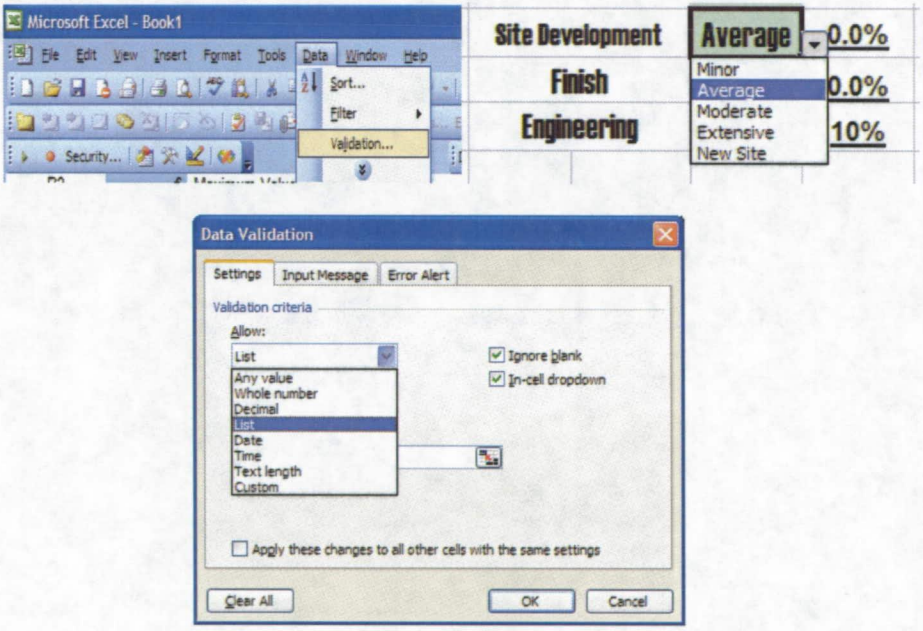


Figure 1. Data Validation

Data Validation provides many features useful to the cost engineer. For example, the user can select an item from a list, which is very helpful when combined with the VLookup function. The list can refer to a cell range defined as `=A10:A50`, or to a Named Range.

Data Validation is also useful in controlling what can be typed into a field, e.g., only allowing dates, numbers, and numbers between certain values.

To protect complex formulas from being accidentally overwritten, the user can select Custom and enter "" in the formula field.

If the user attempts to enter any value other than those allowed by the validation feature, the message shown in Figure 2 will appear. If desired, this message can be customized.

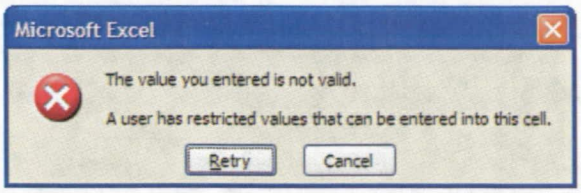


Figure 2. Data Validation Error

If desired, customized messages can provide users with information about certain cells when they are selected, as shown in Figure 3.

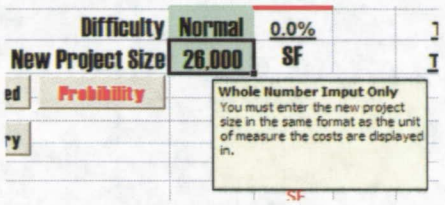


Figure 3. User-Defined Message for Data Validation

Named Ranges

Named Ranges can refer to an individual cell or, more frequently, to a range of cells. A named range can make a complex spreadsheet less confusing. For example, it is easier to understand `=Sum(SF_Costs)` than `=Sum(A5:A16)`. Note that spaces are not allowed in range names. The easiest way to define a range is to select the range, type the name of the range in the Name box, and then press Enter (Figure 4 shows the location of the Name box).

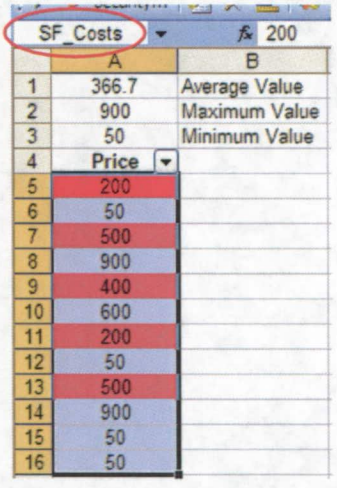


Figure 4. Using the Name Box to Name a Range

Figure 5 shows another way to name a range by using menu commands. Select the range to be named, then select **Insert > Name> Create**, and indicate the portion of the range that should be used as the range name.

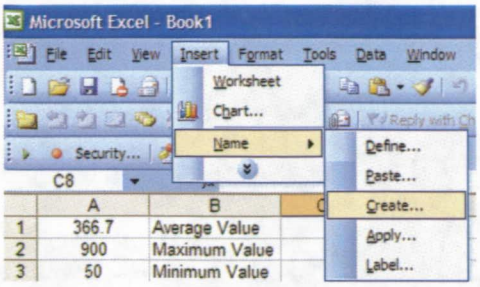


Figure 5. Using Menu Commands to Name a Range

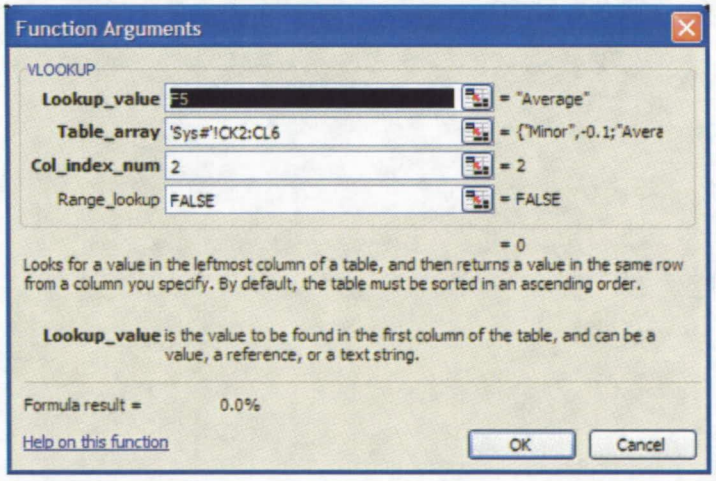
Ranges can be manipulated with several menu functions. They can easily found by setting the zoom on your worksheet to any percentage lower than 40%.

VLookup

The VLookup feature is very helpful in automating spreadsheets. Its uses vary from looking up the correct percentage for our data validation example, to automatically applying escalation to an historical project. At first glance, it looks complicated, but it is actually fairly straightforward. The following functions are available with VLookup:

Lookup_value	Cell value that you want to look up from your list
Table_array	Address of the list that you want to look in
Col_index_num	Column number that the answer will come from
Range_lookup (optional)	True requires an Exact match. False does not require an exact match.

Figure 6 illustrates the dialog box for the VLookup function.



Item to Be Looked up	Lookup Value
Minor	-10%
Average	0%
Moderate	10%
Extensive	20%
New Site	40%

Figure 6. VLookup Function

Concatenate

Cell values can be joined with a formula such as the following: `=A1&A2`. However, if a space is desired between the cell values, use a formula such as: `=A1&" "&A2`. Similarly, a comma can be inserted between the values with a formula such as: `=A1&","A2`.

The Concatenate function is another way to do this. The following is an example of using the Concatenate function.

```
=CONCATENATE($E$14," ",$F$14," ",$D$14," ",$G$14," ",$I$7," ",$J$7)
```

When calculated by Excel, the results would equal:

New Concrete Office Bldg 26000 SF

Figure 7. Concatenate Function

It should be noted that text strings can also be separated in Excel. The formula shown in Figure 8 shows how to remove the first word from a text string.

Mary	Mary had a little lamb
=LEFT(B3,FIND(" ",B3)-1)	Mary had a little lamb

Figure 8. Removing the First Word from a Text String

It is more difficult to remove the last word but it can be done as shown in Figure 9.

Mary had a little lamb	lamb
Mary had a little lamb	=RIGHT(B3,LEN(B3)-FIND("**",SUBSTITUTE(B3," ","**",LEN(B3)-LEN(SUBSTITUTE(B3," ","")))))

Figure 9. Removing the Last Word from a Text String

Go To Special

Excel has a little known capability that can be accessed with the **F5** key. Clicking **F5** will open the **Go To** dialog box, shown in Figure 10. Clicking **F5** and selecting **Special** will open the **Go To Special** dialog box, also shown in Figure 10. This allows the user to locate all worksheet cells that contain the selected criteria. When selected, they can be color-coded for easy identification or can be selected on an individual basis.

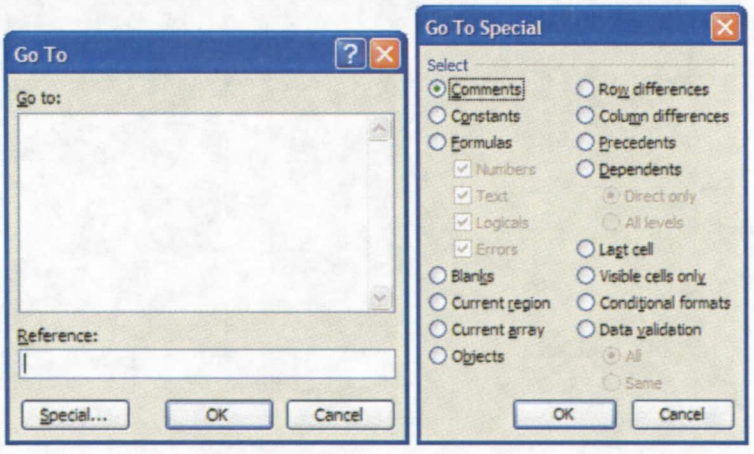


Figure 10. Go To and Go To Special Dialog Boxes

Absolute References

Cell references are typically defined as `=A1*A2`, which works fine unless you want to copy the formula to another location. If this happens, the cell references will change unless Absolute Cell references (shown in Figure 11) are used.

1	=A1	=$A$1	=$A1
2	=A2	=$A$1	=$A2
3	=A3	=$A$1	=$A3
4	=A4	=$A$1	=$A4
5	=A5	=$A$1	=$A5

Figure 11. Absolute References

The following are examples of absolute cell references:

`=$A$1`	Always refers to cell A1 no matter were the reference is copied
`=$A1`	Always refers to Column A, but the row is allowed to shift
`=A$1`	Always refers to Row 1, but the column is allowed to shift

Reveal Formulas

All of the formulas on a worksheet can be revealed by pressing Ctrl `. The sheet will revert back when you press Ctrl ` again.

Formula Descriptions

A typical Excel formula, for example `=Average(A10:A100)*A1`, gives no indication of what the formula does. However, this can be easily remedied. Formula descriptions can be easily added if the proper format is used. The following is an example of the proper format: `=Average(A10:A100)+N("Historical SF Costs")*A1+N("New Project Size")`. The *N* function will return a value of 0 for any text entered, so it does not interfere with this calculation.

Automatically Change Chart Titles

Excel has a little known capability to automatically update chart labels and drawing objects. This is easy but not very intuitive. Select the item and enter = in the formula bar followed by the address of the cell with the desired content. In the example shown in Figure 12, entering `=B1` places the content of cell B1 in the chart title and in the drawing object.

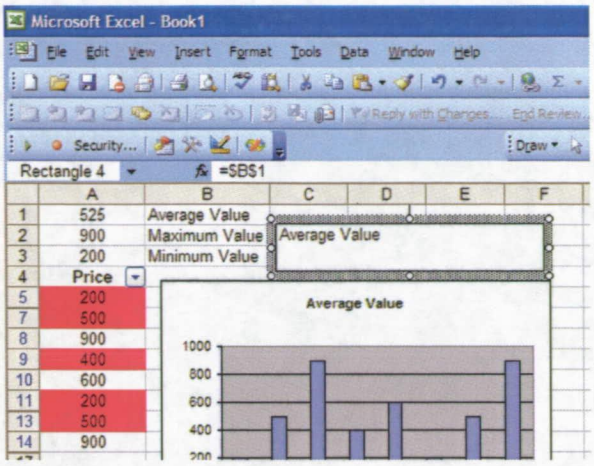


Figure 12. Automatically Updated Chart Titles

If Function

The IF function can be helpful in many chores. One very useful feature is its ability to eliminate the annoying #Ref, #Value, or #Div/0 error messages. This is done as follows:

```
= IF(Test Condition, Instruction if test is True,
Instruction if test is False)
```

or

```
=IF(A1>1, A1/B1,0)
```

Conditional Formatting

The conditional formatting feature, used for identifying information in a spreadsheet, is available from the Format menu as shown in Figure 13.

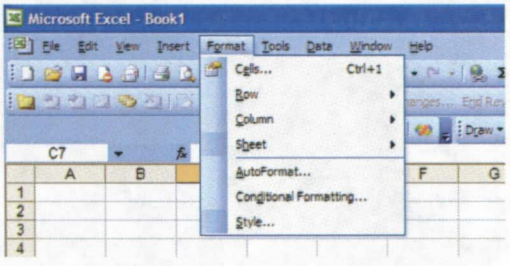


Figure 13. Accessing Conditional Formatting

The conditional formatting dialog box allows the user to input up to three conditions, as shown in Figure 14. Formatting only has to be applied to one cell. It can then be copied to other cells using the Copy-Paste function or the Format Painter.

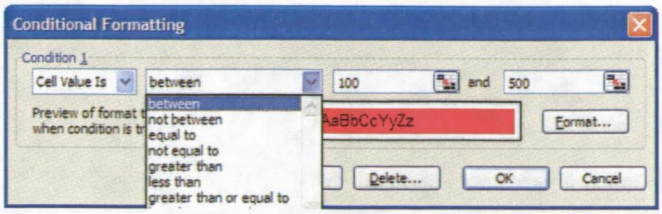


Figure 14. Conditional Format Choices

The results of the conditional formatting are shown in Figure 15. Note that this tool only affects the cell's format (color, font size, font type, or border), not the cell's value. But it can be extremely useful in examining large spreadsheets.

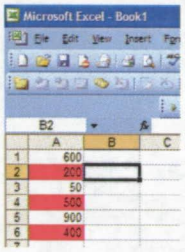


Figure 15. Results of Conditional Formatting

AutoFilter

The AutoFilter tool is accessed by the Data menu as shown in Figure 16. It offers a dropdown window that shows all cell values in ascending order. Whatever value the user selects will be the only value shown. AutoFilter also offers the ability to make a custom selection, when can be very handy when working with large data sets.

In the example shown in Figure 16, the custom AutoFilter will filter the list shown and only reveal items greater than or equal to 200.

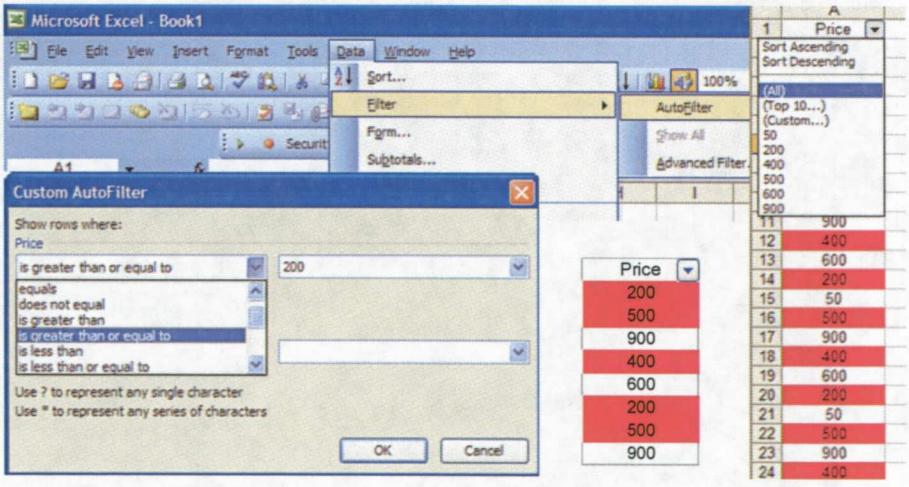


Figure 16. Custom AutoFilter

Subtotal

The Subtotal command, when combined with the AutoFilter, is a very powerful feature that you will find yourself using frequently. By inserting three blank rows at the top of your spreadsheet and using the Subtotal feature, you can easily obtain a great deal of information about your data.

As shown in Figure 17, **Function_num** is a number from 1 to 11 (including hidden values) or 101 to 111 (ignoring hidden values) that specifies which function to use in calculating subtotals within a list.

525	Average Value
900	Maximum Value
200	Minimum Value
Price	
200	
500	
900	
400	
600	
200	
500	
900	

Function_num

1 AVERAGE
2 COUNT
3 COUNTA
4 MAX
5 MIN
6 PRODUCT
7 STDEV
8 STDEVP
9 SUM
10 VAR
11 VARP

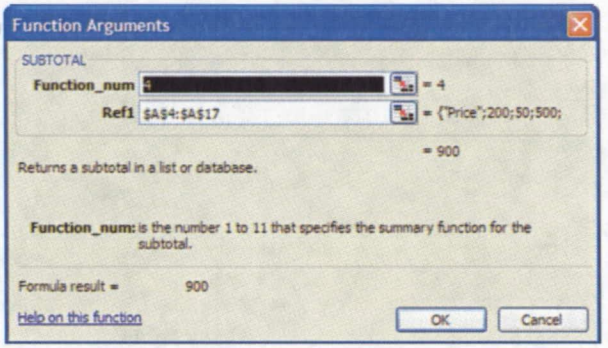


Figure 17. Subtotal Function

If there are other subtotals within ref1, ref2, etc. (or nested subtotals), these nested subtotals are ignored to avoid double-counting.

The Subtotal function ignores any rows that are not included in the result of a filter, no matter which **Function_num** value you use.

The Subtotal function is designed for columns of data, or vertical ranges. It is not designed for rows of data, or horizontal ranges.

Conversions

Excel offers a built-in unit-of-measurement conversion—the "add in" function. It allows units to be converted into values for weight/mass, distance, time, pressure, force, energy, power, temperature, and liquid measure. The Analysis ToolPak must be "added in" to use this feature. See the section on Using Add-In Programs for more information and instructions.

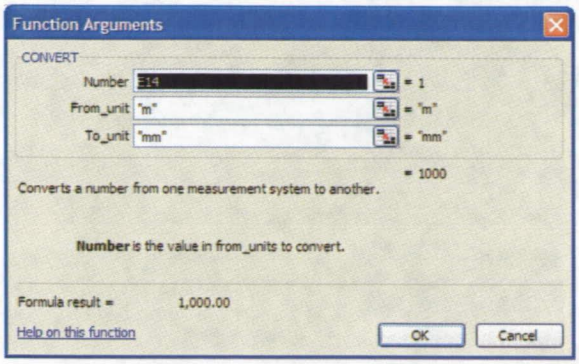


Figure 18. Unit-of-Measurement Conversions

Reducing File Size

Excel files that are used routinely can become huge. Frequently, this is a result of residual formatting in empty cells. A good way to reduce file size is to remove unnecessary formatting by highlighting the first empty row and clicking **Ctrl + Shift + Down Arrow**, then selecting **Edit > Clear All**. This removes all unused formatting and can substantially reduce file size.

Adding Custom Functions to Excel

It is astonishingly easy to add custom functions to Excel by using VBA commands. These commands function exactly like Excel's built-in commands. Numerous websites offer custom VBA code at modest prices for those too timid to try writing their own. Other websites, such as Rentacoder.com code, enable you to offer work for programmers to bid on. In using VBA function, the first step is to access the Visual Basic menu bar as shown in Figure 19. Note that the Macro Security setting (available from the Tools menu) must be set to "medium" for any VBA function to work.

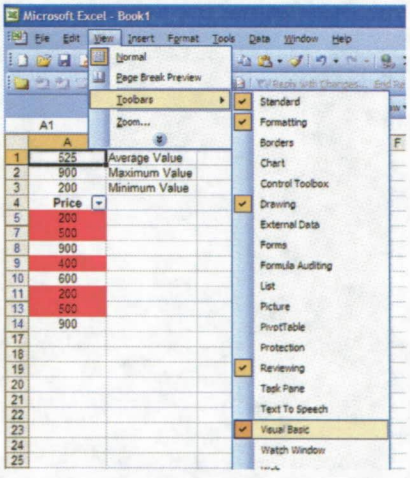


Figure 19. Accessing the VBA Menu

Then next step, shown in Figure 20, is to select the Visual Basic Editor, which will open up a new program.

Figure 20. VBA Menu Bar

When the new program is opened, a module must be inserted by selecting **Insert > Module** as shown in Figure 20.

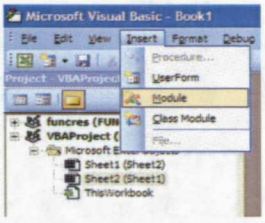


Figure 21. Insert Module

When a module is inserted, a blank window appears. The following commands must be typed in this window, with the underlined items entered exactly as shown. Spaces are not allowed between words connected with _. Any word followed by "`" is ignored by VBA, so those symbols are used for documenting the functions.

```
Function New_Function (User_Variable)

`This custom function multiplies a user variable by 10 and
returns the answer in the cell containing the new function.

New_Function = 10 * User_Variable

End Function
```

Of course, this example is rudimentary; much more complex functions can be used. The example function can be made more user-friendly by adding help that will be displayed on the menu bar. To do this, the following set of instructions must be added to our VBA module:

```
Sub setoptions()

Application.MacroOptions Macro:="New_Function", _

Description:=" This function multiplies a user variable by 10
and returns the answer in the cell containing the new function "

End Sub
```

After including the help instructions in our custom function, you must initialize them by placing the cursor on the word "Description" and clicking on the green arrow (see Figure 22).

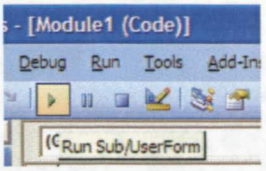


Figure 22. Initializing Help

Figure 23 illustrates the custom code for the new function in the example.

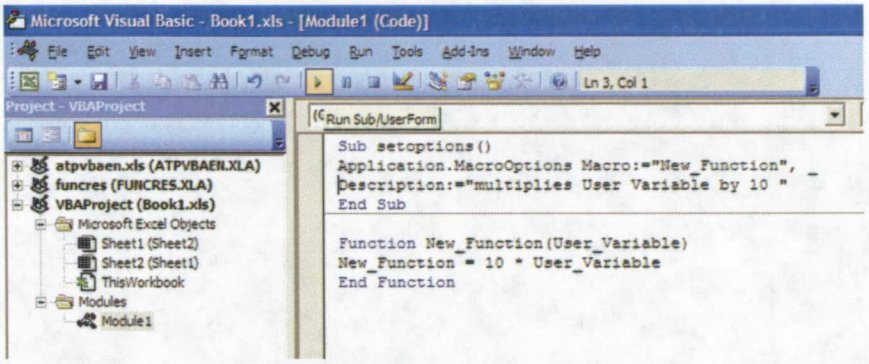


Figure 23. VBA Code for Custom Function

Figure 23 shows the result of the completed custom function.

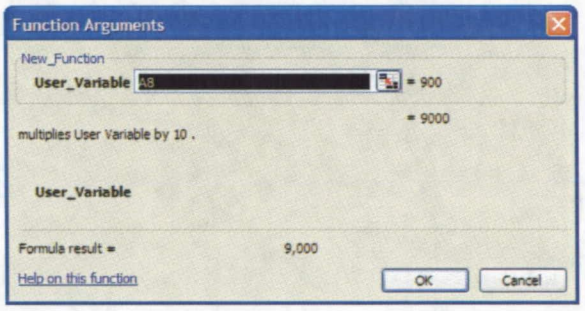


Figure 24. Completed Custom Function

Custom functions, as shown, only work in the spreadsheet in which they are created. Whoever opens the spreadsheet can use them. However, if the spreadsheet is saved as an "add-in" by using **File > Save As** (see Figure 25), they can be used in any spreadsheet opened by the user. The custom functions will not be available when the file is opened by others unless they have also installed the add-in.

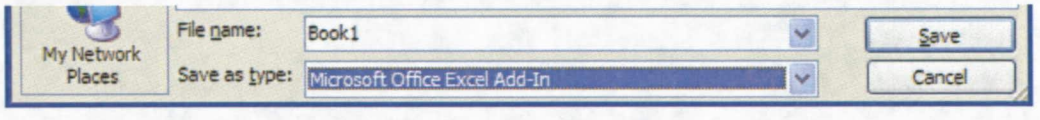


Figure 25. Add-In Feature

Using Add-In Programs

To work, add-ins must be installed after they are created. This is a straightforward process. The add-ins menu has a Browse function that will allow you to locate your new

add-in. Once located, it just has to be checked, as shown in Figure 26. Once checked, the custom functions will be available in any spreadsheet that you open.

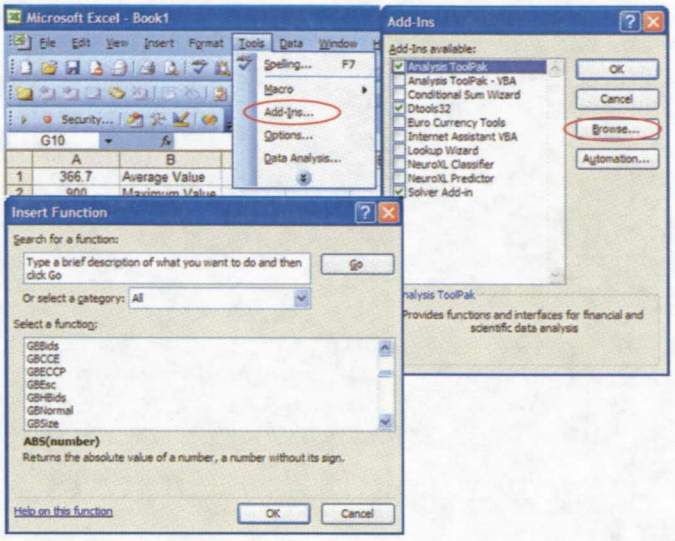


Figure 26. Add-In Installation

Examples of Custom Functions

The following sections contain examples of custom functions used by the author.

Number of Bids New Project = GBBids

```
Function GBBids(Number_of_Bids) As Double
'Number of Bidders for New Project
'ALG 1 y = 1.2686x^-0.1218
    GBBids = 1.2686 * Number_of_Bids ^ -0.1218
End Function
```

Number of Bids Historical Project = GBHBids

```
Function GBHBids(Number_of_Bids) As Double
'Number of Bidders for Historical Project
'ALG 2 y = .74x^0.14
    GBHBids = 0.74 * Number_of_Bids ^ 0.14
End Function
```

Size = GBSize

```
Function GBSize(Historical_Size, New_Size) As Double
    GBSize = 1.010001 * (New_Size / Historical_Size) ^ -0.101
End Function
```

Normalize = GBNormal

```
Function GBNormal(Number_of_Bids, Historical_Size, New_Size,
Orginal_Cost) As Double
    GBNormal = (0.7598 * Number_of_Bids ^ 0.125) * Orginal_Cost
* (1.010001 * (New_Size / Historical_Size) ^ -0.101)
End Function
```

CCE from ECCP = GBCCE

```
Function GBCCE(ECCP) As Double
GBCCE = ((ECCP * 1.005) * 1.1) * 1.1
```

```
End Function
```

ECCP from Unit Cost = GBECCP

```
Function GBECCP(Unit_Cost) As Double
GBECCP = (((Unit_Cost * 1.15) * 1.1) * 1.1) * 1.015
End Function
```

Escalation = GBEsc

```
Function GBEsc(Percent_YEAR, Number_Years, Cost) As Double
GBEsc = ((((Percent_YEAR / 100) + 1) ^ Number_Years)) * Cost
End Function
```

VBA Module Naming

By default, Excel assigns numbers to identify any new modules. However, these numbers are not very descriptive; and on a large worksheet, it can be difficult to find the VBA code that you want to modify. The best way to address this is to rename the modules in descriptive terms. To do this, select the Properties window and then change the name as shown in Figure 27. This name cannot be the same as the name for the macro contained in the module, or the macro will not run.

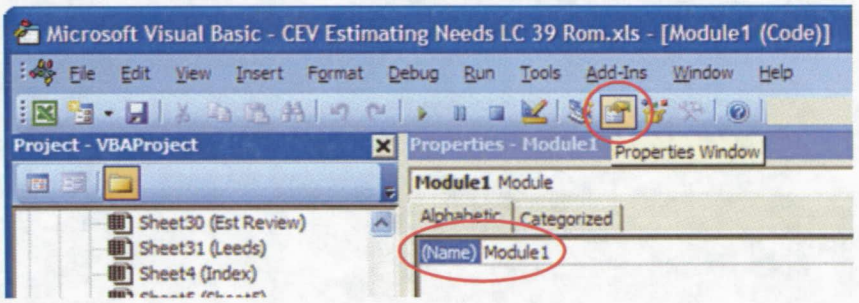
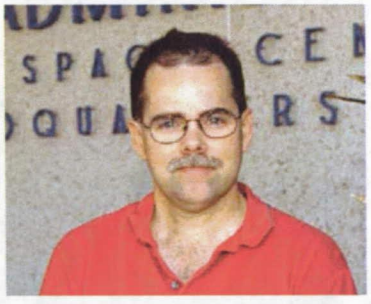


Figure 27. Naming VBA Modules



Mr. Glenn C. Butts, CCC
Program Analyst
NASA
DX-D, Bldg. M6-0399
Kennedy Space Center, FL 32899
Phone 321-867-7198
Email: glenn.c.butts@nasa.gov

Suggested Reading

1. Jelen, Bill 2004, "VBA and Macros for Microsoft Excel", *Que,* Indianapolis, Indiana

2. Jelen, Bill and Joseph Ruben, 2003, "Mr. Excel On Excel," *Holy Macro! Books,* Uniontown, Ohio